**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**DIGITAL SYSTEM DESIGN**

**ASSIGNMENT 02**

**SUBMITTED BY:** AIMUN TARIQ

CE-38-A

**SUBMITTED TO:** DR. SAJID GUL KHAWAJA

QUESTION NO 01;

module tb\_fa();

reg a, b, Cin;

wire sum, Cout;

full\_adder obj ( .a(a), .b(b), .Cin(Cin), .sum(sum), .Cout(Cout));

initial begin

//Inputs

a = 0; b = 0; Cin = 0; #100;

a = 0; b = 0; Cin = 1; #100;

a = 0; b = 1; Cin = 0; #100;

a = 0; b = 1; Cin = 1; #100;

a = 1; b = 0; Cin = 0; #100;

a = 1; b = 0; Cin = 1; #100;

a = 1; b = 1; Cin = 0; #100;

a = 1; b = 1; Cin = 1; #100;

end

endmodule

module tb\_rca ();

reg [3:0] a, b;

wire [3:0] sum;

wire Cout;

integer i,j;

ripple\_adder obj\_01 ( .a(a), .b(b), .sum(sum), .Cout(Cout));

initial begin

// Inputs

a = 0; b = 0;

#50

for (i = 0; i < 16; i=i+1) //a

begin

for (j = 0; j < 16; j=j+1) //b

begin

a = i; b = j; #10;

end // loop 02

end // loop 01

end // initial

endmodule // ripple\_carry\_adder\_tb

module full\_adder(a, b, Cin, sum, Cout);

input a, b, Cin;

output sum, Cout;

reg sum, Cout, t1, t2, t3, t4, t5, t6;

always @(\*)

begin

t1 = a ^ b;

t2 = a & b;

t3 = ~t2;

t4 = t1 & Cin;

t5 = ~t4;

sum = t1 ^ Cin;

t6 = t3 & t5;

Cout = ~t6;

end

endmodule

module ripple\_adder(a, b, sum, Cout);

input [3:0] a, b;// Two 4-bit inputs

output [3:0] sum;

output Cout;

wire w1, w2, w3;

full\_adder obj\_01(a[0], b[0], 1'b0, sum[0], w1);

full\_adder obj\_02(a[1], b[1], w1, sum[1], w2);

full\_adder obj\_03(a[2], b[2], w2, sum[2], w3);

full\_adder obj\_04(a[3], b[3], w3, sum[3], Cout);

endmodule

QUESTION NO 02;

module tb\_MAC();

reg clk, rst\_n;

reg [7:0] a, b, c, d;

wire Acc;

integer i, j, k, l;

MAC obj ( .a(a), .b(b), .c(c), .d(d), .Acc(Acc), .clk(clk), .rst\_n(rst\_n));

initial begin

//Inputs

clk = 0; rst\_n = 0;

a = 0; b = 0; c = 0; d = 0;

#50

for (i = 0; i < 256; i=i+1) //a

begin

for (j = 0; j < 256; j=j+1) //b

begin

for (k = 0; k < 256; k=k+1) //c

begin

for (l = 0; l < 256; l=l+1) //d

begin

a = i; b = j; c = k; d = l; #2;

end // loop 04

end // loop 03

end // loop 02

end // loop 01

end // initial

always

#5 clk = !clk;

always

#10 rst\_n = !rst\_n;

endmodule

module MAC (a, b, c, d, Acc, clk, rst\_n);

input clk, rst\_n;

input [7:0] a, b, c, d;

output [31:0] Acc;

wire [7:0]t1, t2;

reg [31:0] Acc;

assign t1 = a \* b;

assign t2 = c \* d;

always @(posedge clk, negedge rst\_n)

begin

if (rst\_n == 0)

begin

Acc <= 0;

end

else begin

Acc <= t1 + t2 + Acc;

end

end

endmodule

QUESTION NO 03;

module tb\_t03();

reg clk, rst\_n, sel;

reg [2:0] x1, x2, a1, a2, a3;

wire [2:0] Y;

integer i, j, k, l, m;

task\_03 obj\_01 (.x1(x1), .x2(x2), .a1(a1), .a2(a2), .a3(a3), .clk(clk), .rst\_n(rst\_n), .Y(Y), .sel(sel));

initial begin

clk = 0; rst\_n = 0; sel = 0;

x1 = 0; x2 = 0; a1 = 0; a2 = 0; a3 = 0;

#10

for (i = 0; i < 8; i=i+1) //x1

begin

for (j = 0; j < 8; j=j+1) //x2

begin

for (k = 0; k < 8; k=k+1) //a1

begin

for (l = 0; l < 8; l=l+1) //a2

begin

for (m = 0; m < 8; m=m+1) //a3

begin

x1 = i; x2 = j; a1 = k; a2 = l; a3 = m; #10;

end // loop 05

end // loop 04

end // loop 03

end // loop 02

end // loop 01

end // initial

always

#5 clk = !clk;

always

#10 rst\_n = !rst\_n;

always

#20 sel = !sel;

endmodule

module mux2to1(a, b, sel, Data\_out);

input [2:0] a;

input [2:0] b;

input sel;

output [2:0] Data\_out;

//Internal variables.

reg [2:0] Data\_out, t1, t2, t3;

always @(a, b, sel, t1, t2, t3)

begin

t1 = a & sel;

t2 = ~sel;

t3 = b & t2;

Data\_out = t1 | t3;

end

endmodule

module task\_03(x1, x2, a1, a2, a3, clk, rst\_n, Y, sel);

input clk, rst\_n, sel;

input [2:0] x1, x2, a1, a2, a3;

output [2:0] Y;

wire [2:0] t1, t2, t4, t5, x3;

reg [2:0] x4, x5;

wire [2:0] t3; wire [5:0] Y;

mux2to1 obj\_01 (.a(x1), .b(x4), .sel(sel), .Data\_out(t1)); // mux 01 ..output = t1

mux2to1 obj\_02 (.a(x2), .b(x5), .sel(sel), .Data\_out(t2)); // mux 02 ..output = t2

mux2to1 obj\_03 (.a(a1), .b(a2), .sel(sel), .Data\_out(t4)); // mux 03 ..output = t4

mux2to1 obj\_04 (.a(a3), .b(x3), .sel(sel), .Data\_out(t5)); // mux 04 ..output = t5

assign t3 = t1 + t2;

assign x3 = t3;

assign Y = t4 \* t5;

always @(posedge clk, negedge rst\_n)

begin

if (rst\_n == 0)

begin

x4 <= 0; x5 <= 0;

end

else begin

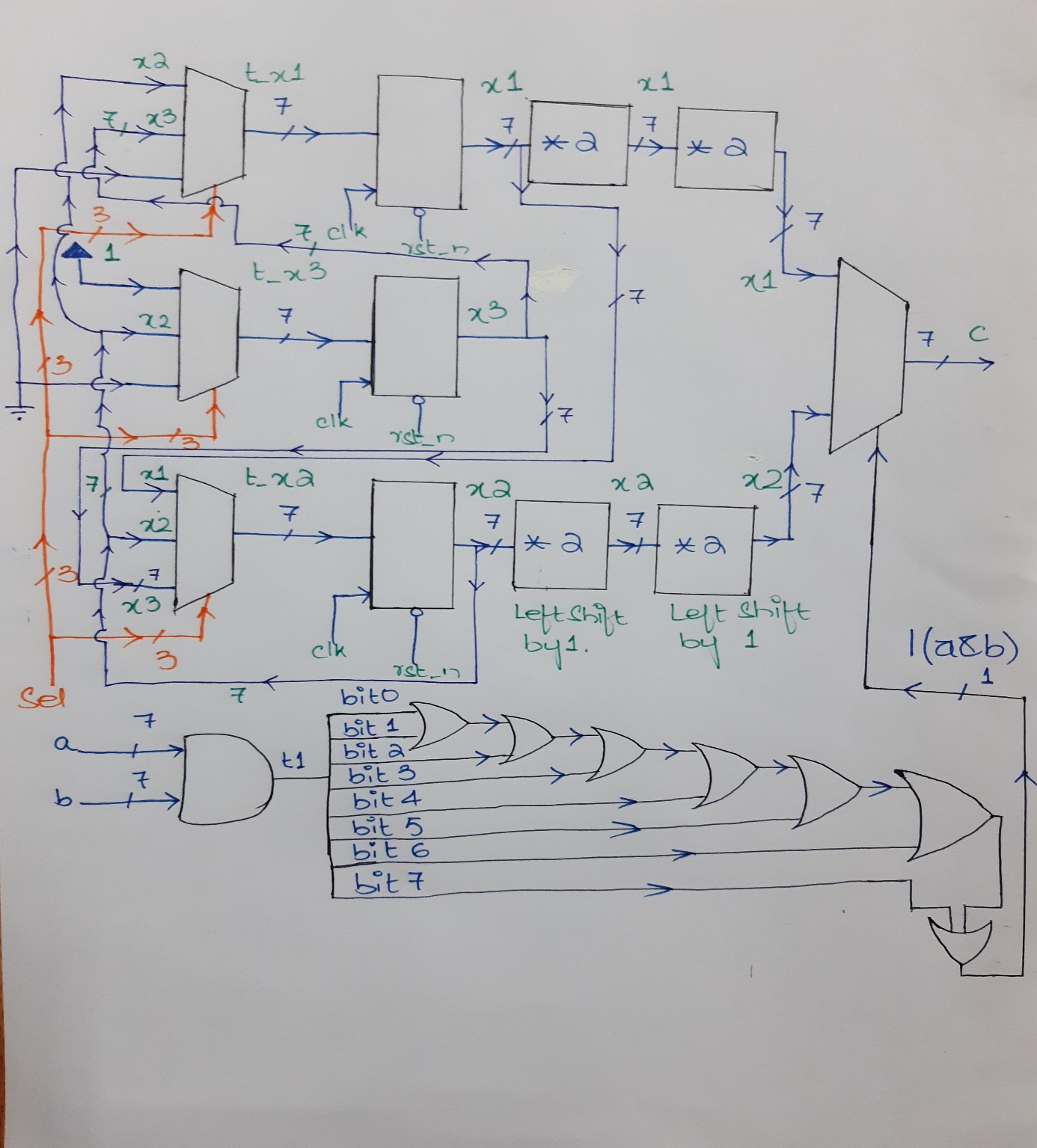
x4 <= t3; x5 <= Y;

end

end

endmodule

QUESTION NO 04;



QUESTION NO 05;

module ALU(a, b, sel, out);

input [3:0] a, b; // ALU 4-bit Inputs

input [2:0] sel; // ALU Selection

output [4:0] out; // ALU 5-bit Output

reg [4:0] out;

always @(\*)

begin

case(sel)

3'b000: // A

out = a;

3'b001: // Addition

out = a + b ;

3'b010: // Subtraction

out = a - b ;

3'b011: // Division

out = a/b;

3'b100: // Remainder

out = a % b;

3'b101: // Logical shift left

out = a<<1;

3'b110: // Logical shift right

out = a>>1;

3'b111: // Greater comparison

out = (a>b)?8'd1:8'd0 ;

default: out = a \* b ;

endcase

end

endmodule